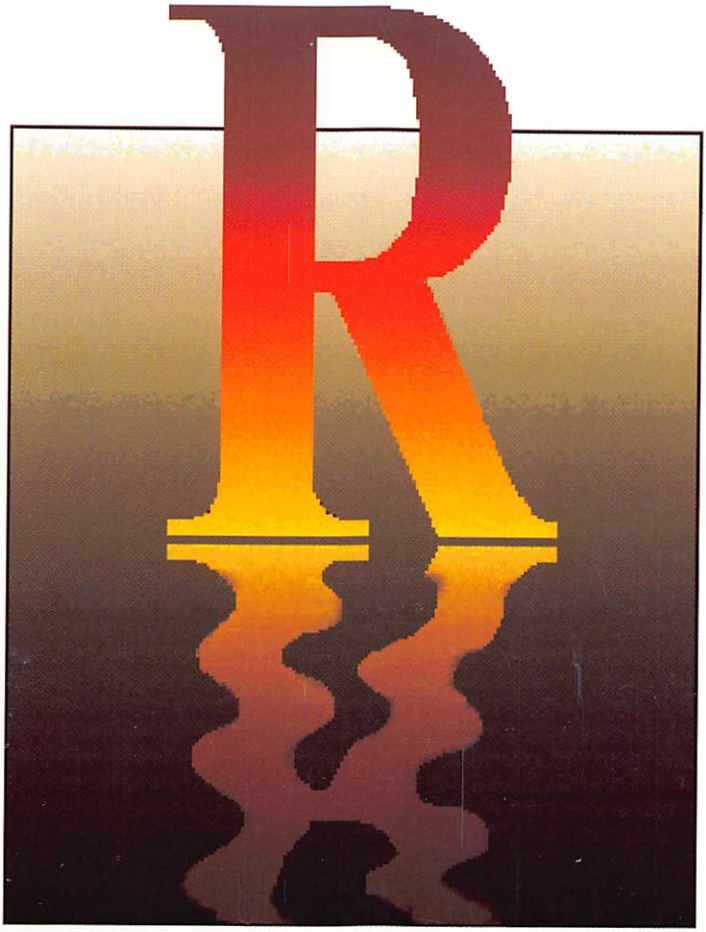

Inst™

GRAPHICS CONVERTER

FOR WINDOWS

GRAPHICS CONVERTER

WINDOWS USER GUIDE

Graphics Converter for Windows is a trademark of IMSI.
Copyright © 1993 International Microcomputer Software, Inc.
All rights reserved.

1938 Fourth Street
San Rafael, CA 94901
Tel (415) 454-7101 • Fax (415) 454-8901
BBS (415) 454-2893

IMSI (UK) Limited
Unit 17 Brook Lane Business Centre
Brentford, Middlesex, TW80PP, United Kingdom
Tel (081) 758-1447 • Fax (081) 758-1667

TurboCAD is a trademark of IMSI

RGC02W 0593

Table of Contents

1 Introduction
2 System Requirements
3 Installation
3 Quick Start
4 Graphic File Formats
4 Classes of Graphic Files
5 Raster Files
5 Vector Files
5 Metafiles
7 The Anatomy of a Graphics File
8 File Data Compression Schemes
12 Raster Formats
12 PCX
17 IMG
18 TIFF
23 Why Tags?
23 TIFF Classes
24 Compression
25 The TIFF Header
28 Organizing Raster Data Under TIFF
29 GIF
29 CUT
30 PIX
31 Vector Formats
31 DXF
32 HPGL
33 EPS
34 GEM
35 CGM
36 Bitmaps
36 Pattern Fills
39 Graphics Converter Error Messages
40 Set Files

41 Operation Tags
41 Generic Format Tags
41 Generic Format Pair Tags
41 Specific Format Tags
42 Operation Tags
43 Generic Format Tags
45 Generic Format Pair Tags
46 Format Codes and Format Specific Tags
46 Vector File Types
47 Raster File Types
48 Mixed Vector/Raster Formats

Introduction

Thank you for choosing IMSI Graphics Converter for Windows. You will find it a valuable tool for changing DXF and HPGL files to PCX, TIF, GIF, IMG, CUT, PIX, GEM, CGM and EPS formats. In a matter of minutes, vector graphic drawings can be converted into a wide variety of vector and bit mapped formats used by popular graphics programs.

Graphics Converter is easy to use, yet impressively powerful. With a single command line it handles all of the details of moving data into its new form. You can also tailor the conversion by including file specifications.

This manual walks you through the basic steps needed to install and run Graphics Converter. You will find the most needed items at the front, with detailed references towards the back.

INSTALLATION walks you though the setup procedure to load Graphics Converter to your hard disk.

QUICKSTART shows you how to convert files with the basic default settings. You will likely find that most, if not all, of your conversions will fall into this category.

GRAPHICS CONVERTER OPTIONS shows how to use parameters to control the conversion process.

GRAPHIC FILE FORMATS is a detailed reference to different graphics formats.

SET FILE explains how to alter the manner in which a specific file is converted.

We feel certain that you will find Graphics Converter for Windows a worthwhile addition to your Windows software library.

System Requirements

- ☐ 286, 386, 486 100% IBM compatible
- ☐ Windows Version 3.0 or greater
- ☐ A hard drive with 2 megabytes of free space
- ☐ Either a 5.25" or 3.5" floppy drive
- ☐ 2 Megabytes of RAM
- ☐ A math co-processor is recommended for DXF conversions

Installation

Graphics Converter for Windows must be installed onto your hard disk. Because the files on the disks are in a compressed format, you must use the SETUP program to load them properly. Transferring files using DOS commands such as COPY or XCOPY will not work correctly.

To begin, you must have Windows running. Place the install disk in drive A (or B) of your computer (if using 5 1/4 disks, close the latch). Under the **File** menu in Windows, select **Run.** This will open a dialog box where you can type:

A:SETUP

(or B:SETUP if using drive B)

The setup program will then transfer the files from the floppy disk to the hard disk. If you are using 5.25" disks, you will be prompted to change disks when necessary.

When finished, the setup program will automatically put a Graphics Converter icon in a new Program Manager group.

Quick Start

To use Graphics Converter for Windows, double click on the program icon.

The Graphics Converter interface is like a normal file selection window. First, you select the source image and then secondly, select the graphic standard into which the image will be converted.

The source file is shown at the top. You may either write the path and name of the file, or select it from the drives and directories in the usual manner.

Next, you must select the graphic standard (e.g. TIFF, GIF, etc) into which the image file will be converted. By pointing at and pressing the down arrow in the "**Convert File To...**" box, you can see a list of the available file standards.

When you have made your selections, either click **OK** or press <Return>.

If you make a mistake or want to exit, click **Cancel** or press <Escape>.

Destination Formats

These are the available formats into which your graphic can be converted. Their extenders are:

PLG	Hewlett Packard HPGL
DXF	Autodesk DXF
PCX	ZSoft PCX
TIF	Tagged Image File Format
GIF	Compuserve GIF
IMG	Digital Research raster
CUT	Dr. Halo CUT
PIX	Inset Systems PIX
GEM	Digital Research vector
CGM	Computer Graphics Metafile
EPS	Encapsulated PostScript

Graphics File Formats

The following information on graphic formats is for general information. If you are having problems with file conversion from different programs on different systems, this may help in spotting the problems and possibly solving them.

Classes of Graphics Files

File formats fall into one of three classes: Raster, Vector, and Metafiles. Despite the fact that there are three classes of files listed above, there are only two ways to create an image on a computer. The first involves the description of the color of each possible dot on a given image. As in the pointillist technique of the Impressionist painters, images are represented (and coded) as a large canvas full of dots. In the computer world, these images are known as Raster images. A non-computer example of a raster image is that of an image created by scores of people seated in a football stadium holding up various colored cards at predetermined intervals. When viewed from the other side of the stadium or from the ubiquitous overhead blimp, the pattern created by the colored cards creates the illusion of an image.

Vector files, on the other hand, contain no information about specific dots. Instead, vector files are an assimilation of mathematically described or coded objects used to create an image. In order for an image stored in a vector file to be viewed, the objects in the list must be *rendered* by a program on an output device. Thus, vector images are not linked to any particular piece of hardware; it is up to the program to properly interpret the file and drive the hardware on which the image is to be displayed or printed.

Raster Files

- ☐ Good for scanned images.
- ☐ Good for 'Paint' packages.
- ☐ Good for random patterns and textures.
- ☐ Size of image depends on viewing device.
- ☐ Scaling or Resizing can affect image quality.
- ☐ File size matches the number of dots and colors in image.
- ☐ Modification based on dot color and location.
- ☐ Printed on devices that print dots.

Vector Files

- ☐ Good for drawing objects such as lines, circles, squares, text, etc.
- ☐ Good for line drawing packages, including TurboCAD.
- ☐ Size of image easily changed to fit the device resolution.
- ☐ Good for charts and graphs.
- ☐ Good for predictable patterns (hatching).
- ☐ File size related to number of objects rather than size of image.
- ☐ Modification based on individual objects present in image.
- ☐ Best printed on line drawing devices.

Metafiles

Finally, there is a class of files that houses everything under one roof. These files are called Metafiles. Metafiles can accommodate both raster and vector information, although software packages very rarely write both types of information into a Metafile. At this point the majority of the Metafiles in existence contain vector information only.

As you might expect, each class has its strengths and weaknesses, which are discussed below.

Summary of File Formats

File Types	Extension	Class	Supported as	Notes
AutoCADDXF	DXF	Vector	Source	ASCII Encoded
HP Plotter	PLT	Vector	Source	Based on HP 7475A plotter
PC Paintbrush	PCX	Raster	Destination	Very Popular
TIFF	TIF	Raster	Destination	Many Flavors
Compuserve	GIF	Raster	Destination	Interlace
GEM Image	IMG	Raster	Destination	GEM Standard
Dr. Halo	CUT	Raster	Destination	Seperate Palette File
Inset Systems	PIX	Raster	Destination	Can include text
GEMMetafile	GEM	Metafile	Destination	Includes GEM 4
PostScript	EPS	Metafile	Destination	Includes Raster-TIFF link for viewing

The Anatomy of a Graphics File

Computer Graphics files typically have two parts. The first part, known as the File Header, contains bookkeeping information about the image as a whole. The second part, known as the Image Data, contains the actual dot or object information that makes up a particular image. A simple file may be comprised of the following information:

- Version
- Length of Header
- Image Type ID
- Color Scheme
- Image Width
- Image Height
- Image Data

If this were a *raster* image, the *image data* would simply be a long list of numbers in which each number represents a color value. The application program would be responsible for properly formatting the long string of dots into rows (based on the Image Width) so that the image appears as expected when displayed. Raster images contain explicit descriptions for each dot, including sizing, scaling, color manipulation, and position.

If the following were a *vector* image, the *image data* would be a list of object descriptions, for example:

```
Line -3,-4,3,4; Circle 0,0,5; ...
```

Let's assume that Fred wrote a vector graphics program. The application program, knowing it is dealing with Fred's Universal Vector Format, knows that the first two numbers following a Line command represent the coordinates of one end of the line and the next two, those for the other end. To round out our example, the first two numbers following a circle command represent the circle's center point coordinates and the third number represents its radius. Any application program which supports Fred's Universal Vector Format has to abide by these rules or else incur the wrath

not only of Fred, but more importantly, all of Fred's customers who expect their images to look exactly the way they did when they were created.

The major tradeoff in file format design is between flexibility and simplicity. Designers, creating robust formats addressing specific technical problems relating to hardware, focus on accommodating the needs of their environment, not the general needs of the computer graphics world. What is intended to be a flexible and powerful standard often becomes a nightmare of incompatibility that few programs attempt to read and write properly or completely. Typically, the more flexible and complex a file format is, the more programmers tend to cut corners when creating files.

File Data Compression Schemes

Since graphics files, especially raster files, historically required more storage space than their alphanumeric counterparts, various techniques were used to reduce their size. Such data compression techniques, normally applied to the data portion of raster files, range from simple run length encoding to the complex dynamically indexed data style used by TIFF. Regardless of their individual complexity, these techniques all work on the principle of representing repeating data strings with codes (tags) that are either stored in a table or generated on the fly, and since this is the case, the degree of success achieved by a given compression technique depends greatly on the content of the particular image file. The more common data compression schemes are described in general terms below.

- ☐ Run Length Encoding — Run length encoding (RLE) is the simplest form of data compression. RLE is a way to avoid repeating consecutive occurrences of identical data items. For example, if you have an image consisting of a blue rectangle, an uncompressed data file would contain a data item (whatever the code was for blue) for each pixel. (For an EGA screen this would be 640x350 or 224,000 identical data entries.) Using RLE, this value could be dramatically reduced by rephrasing the description to read "Use the blue value 224,000 times". Thus, the

data portion would only consist of a tag to indicate that RLE was being used, the code for the color blue, and the number 224,000 to indicate the number of consecutive instances of blue. RLE can be used with pattern runs as well as solid color runs.

- ☐ Vertical Replication — This technique is used by some file formats (WordPerfect WPG and GEM IMG) to avoid having to describe identical horizontal rows more than once. It is similar to RLE in that the data item (one horizontal row of the image) is described once and then repeated a number of times.

- ☐ Keyed Compression Techniques — There are variations on the RLE theme that use sets of tags (tables) for predefined patterns to achieve data compression. Most notably in the graphics area is the use of the Huffman encoding scheme used by the CCITT Group 3 FAX format. The Huffman tables are optimized for facsimilie images and produce good reliable results in that area. Since Group 3 is designed around the Huffman tables and any program or device supporting Group 3 images must contain these tables, the image files do not carry the Huffman table information. The down side of using a fixed table encoding scheme is that it is not sensitive to the content of the image. Therefore, some files compress extremely well while others do not.

LZW Encoding — LZW compression, named after the two individuals who invented it (Lempel-Ziv and Welch), is one of the more clever (and complex) forms of data compression used in PC based graphics files. The LZW method not only generates a content sensitive compression table on the fly, but also is designed so that the table itself does not need to be stored with each image!

The LZW process works something like long division with a twist. In long division, you start with a two strings of digits — the dividend and the divisor, you generate a bunch of intermediate numbers, and end up with another string of digits. The long division process works something like this:

1. Divide the first digit of the dividend by the divisor.

2. Save the resulting number (R1)

3. Multiply R1 by the divisor and subtract the resulting digits from the appropriate digits of the dividend. Append the next dividend digit in line to this remainder and use it as the new dividend. Then repeat the process.

Thus, the general method involves a testing operation, the result of which is not only retained as new information, but also recycled back into the next testing operation. When you are finished, all the intermediate numbers as well as the original dividend can be discarded since you can regenerate them just by knowing the divisor and the quotient.

In LZW, the role of the divisor and intermediate numbers are played by the dynamic table, the dividend is the file to be compressed and the quotient is resulting compressed file. The key to understanding the process is understanding how the table is built. Just as with long division, where you start with a known divisor, in LZW you start with a known table. The size of this table depends only on the number of possible values for each data item. Thus, for a 16 color file, your initial table would contain 16 entries.

The process works something like this:

Starting at the beginning of the data block with an empty string variable (DC for Data Current)...

4. Get the next data item (DI) from the data block.

5. Check to see if the concatenation of DC and DI is in the table.

   If yes, then assign this concatenation to DC (DC= DC+ DI). Go back to Step #1.

   If no, then write the table entry number for DC out to the file,

add (DC+ DI) to the table, and assign DI to DC. Go back to Step #1.

Thus, the space savings is achieved by virtue of the fact that rather than writing out strings of information to the file, you are writing out table entry numbers to represent those strings. The table is built based on a testing operation, the result of which is not only saved as new information, but is also recycled back into the testing procedure to help build the intermediate table. This description is only intended to describe the heart of LZW compression algorithm in relation to other forms of data compression. An additional layer of complexity (and space savings) is added by the fact that all strings are written to the file on a bit basis.

Raster Formats

PCX

The PC Paintbrush Format	ZSoft Corporation 450 Franklin Road,, Suite 100 Marietta, GA 30067 USA Tel.404-428-0008
Vital Statistics:	Raster type 64K by 64K image size (Max) 256 colors (Max) Flexible palette PCX, PCC extensions

PCX is one of the oldest and most common raster formats available on PC's. It is simple to read and write, and most graphics and desktop publishing programs that import raster graphics support the PCX format. Since it has been around for such a long time, there have been many iterations of the PCX format. Those most commonly used are listed below:

- ☐ Version 2.5 — This version does not support a flexible palette. The standard EGA palette is used.
- ☐ Version 2.8 — This version supports a modifiable palette for EGA and VGA 16 color files.
- ☐ Version 3.0 — This version includes support for 256 color images. The palette describing the RGB content of those 256 colors is appended to the end of the bitmap data. Refer to the notes at the end of the header description.

The PCX Header		
Byte #	Description	Notes and Values
0	Manufacturer ID	Always set to 0AH
1	Version Number	0 = Version 2.5
		1 = Version 2.8
		3 = Version 2.8 w/o Palette
		5 = Version 3.0
2	Encoding Technique	Should always be set to 1 Simple RLE. All other values are non-standard.
3	Bits/Plane/Pixel	Use with Byte 65 (# of Planes). Valid entries are 1, 2, or 8. Refer to Note 2.
4 — 5	Window—Xmin	These two byte coordinate descriptions are never negative values and describe a window that includes the coordinates themselves.
6 — 7	Window—Ymin	
8 — 9	Window—Xmax	
10 — 11	Window—Ymax	
12 — 13	Horizontal Resolution	Measured in dots per inch (dpi). 2 byte limit.
14 — 15	Vertical Resolution	
16 — 63	Color Map	Colors specified in RGB triplets, 1 byte per R, G, and B. If fewer than 16 colors are used, pad with zeros.
64	Reserved	Always set to 0.
65	# of Bit Planes	Ties in with Byte #3. Refer to Note 2 below.
66 — 67	Bytes/Scanline/Plane	Also known as image width. If you're writing a PCX viewer, you are responsible for adhering to width described in bytes 4 —11. This value must be aligned on even byte boundaries. Pad to edges with zeros to avoid artifacts.
68 — 69	Palette Information	1 = Color or B&W 2 = Gray Scale Generally ignored.
70 — 127	58 Bytes of Padding	No function. Always fill with zeros.

128	Beginning of Data Area PCX Compression Map	Calculate length from DOS file length. See Notes 4 & 5 below.
	256 Color Palette Info Appended at (EOF — 768) for some V3.0 files.	See Note 4 below.

Notes

[1] All bytes are read in sequential order. Words (byte pairs) are *not* inverted and are read in low byte, high byte.

[2] Byte #3 and Byte #65 must be used together to completely determine the color resolution of the image. Check Byte #65 first to determine the number of planes. If byte #65 is a one (1), then Byte #3 describes the color resolution explicitly. If Byte #65 is a four (4), then Byte #3 should be a one (1), meaning there are four bits per pixel or 16 possible colors.

This anomaly in requiring a specification for Bits per Pixel per Plane arises from the manner in which EGA hardware registers are driven. It is more efficient to store the data grouped in planes rather than pixels. In other words, even though each pixel can be one of 16 colors (4-bits), the file stores all the first bit plane information, then all the second bit plane information ..., rather than storing all four bits of the first pixel, then all four bits of the second pixel ...

Byte #65	Byte #3	Mode
1	1	Monochrome or Bi-level (CGA-mono, EGA-mono or other, e.g., Hercules)
1	2	4 Color CGA
4	1	16 Color or Gray EGA, VGA
1	8	256 Color or Gray Extended VGA

3 Since the dimensions are described by 2 byte values, the maximum height and width of any image is 65,535 pixels. Your image width must be aligned on even byte boundaries. Bytes 4 — 11 specify the image width, and bytes 66 — 67 specify the number of pixels per scan line and must fall on even byte boundaries. If you are writing PCX files, pad your scan lines with zeros to insure that no edge artifacts are introduced into your images by PCX readers that don't do this for you. (These artifacts are often seen when PCX files are imported into Ventura Publisher — a vertical strip shows up at the right edge of your image.)

4 In order to accommodate images with 256 colors, PCX files append color palette information to the end of the file. If the version number is 3.0 (byte # 1 set to 5), bits/pixel/plane equals 8 (byte #3 set to 8), and for completeness, the number of planes equals 1 (byte #65 set to 1), then you are *probably* dealing with a 256 color PCX file! In order to translate those pixel color values from 0 to 255 into RGB terms, the 256 color palette is included at the end of the file. There is one byte per R,G, and B for each of the 256 values. Thus the palette contains 768 (3 x 256) entries, grouped as RGB triplets. To be sure that this is in fact a 256 color file, you must seek back 769 bytes from the end of the file and check to make sure that that byte is set to C0 Hex. If so, then the next byte is the beginning of the 256 color palette table. Ignore any information in bytes 16 through 63. Quite QWERTY , but it works.

If you're writing out such a PCX file, append a C0 hex to your bitmap data and then append the 768 byte color palette.

5 The bitmapped data portion of a PCX file uses a simple RLE technique to achieve some disk space savings. The RLE indicator tag is used when the two high bits of a byte are set to 11 (binary). If the two high order bits are set, then the 6 remaining bits specify the length of the run and the next byte specifies the

data to be repeated.

This technique works quite well, but there is some loss for cases in which the image data has individual bytes with a value greater than BF Hex. Such items are specified as having a run length of 1 and consequently require two bytes of storage for one byte of information.

Finally, since the two high order bits are used to indicate RLE, the maximum length for any run is 3F.

PCX Compression Map Examples (All Values in Hex):

Compressed String	Meaning
FF 4E	Strip off C0 from FF to get a run length of 3F (63 Decimal); the run to be repeated is 4E.
C1 FF	The Data Value FF occurs once.
FF 4E B0 43 C1 FF	A run of 63 instances of the value 4E, followed by single instances of the values B0, 43, and FF.

IMG

The GEM IMG Format:	Digital Research Inc. P.O. Box DRI Monterey,, CA 93942 USA Tel.408-649-3896
Vital Statistics:	Raster type 64K by 64K image size (Max) 16 colors (Max) Fixed 16 color palette IMG extension

The Graphical Environment Manager (GEM), created by Digital Research, is an operating environment (or operating system) like Windows, OS/2, or Apple Macintosh Finder. Its native raster format is the IMG format, which means that application programs using raster files written for GEM should support the use of GEM files. Most notably, Ventura Publisher uses the IMG format as its internal format for raster images. Raster images imported into Ventura Publisher are converted to the IMG format.

The IMG Header		
Byte #	**Description**	**Notes and Values**
0 – 1	Image file Version #	Always set to 1.
2 – 3	Header length in *words.*	Most programs create an eight *word* header. Ventura Publisher often creates a nine word header.
4 – 5	# of Planes	1 — Monochrome 2 — 16 colors See Note 2 below.
6 – 7	Pattern Length (for compression)	Can range from 1 – 8. Default value is 2 bytes.
8 – 9	Pixel Width	Microns
10 – 11	Pixel Height	Microns
12 – 13	Image Width	Pixels (65535 limit)
14 – 15	Image Height	Pixels (65535 limit)
16 – EOF	Compressed Bitmap	Refer to description below.

1 The IMG header is word oriented rather than byte oriented. These words are inverted byte pairs and are to be read High Byte, Low Byte.

2 Graphics Converter currently supports only monochrome and 16 color IMG files. Ventura Publisher creates non-standard IMG files when you import image files containing more than 16 colors or gray values. You should not attempt to convert these IMG files with Graphics Converter. If you need to convert such images, use the TIFF or PCX file as your source. If you only have the EPS file to work with, you'll need to use a PostScript conversion utility such as GoScript, UltraScript, or Freedom of the Press.

In the 16 color mode, IMG files always use the standard EGA palette. GEM Paint clips images that it cannot load completely into memory. Beware of this with large image files.

3 Most IMG files use a 2 byte pattern length. All pattern runs found in the file must contain bit patterns of this length. Longer pattern lengths provide more compression but can't be used as often since longer patterns tend to occur less frequently.

4 Pixel Width and Height are given in microns or millionths of meters. (1 micron equals 10^{-6} meters.) These values are seldom used since graphics adapters driving displays specify resolution in dots and printers specify resolution in dots per inch. Generally, raster image size is calculated from the number of dots in an image. A nominal pixel size is 85 microns.

5 Since IMG files are plane oriented, the compression scheme is bit based rather than byte based. The planar approach lends itself well to treating all pixels as either ON or OFF, since colors are broken down into planes. Thus, if two consecutive pixels are different colors, those planes having common ON-OFF values can still be compressed on a bit basis. For example, if we have a row of eight pixels alternating with colors 12 (1100 Binary) and 14 (1110 Binary), the nonplanar binary pattern representing those sixteen pixels would be as follows:

8 pixels of alternating colors								
Decimal	12	14	13	12	14	13	12	14
Binary	1100	1110	1101	1100	1110	1101	1100	1110

The planar approach takes the first bit from each group, then the second, then the third and finally the last, to yield the following:

The same 8 pixels in a plane ordered representation								
Binary	1111	1111	1111	1111	0100	1001	0010	0100

Thus, the data compresses down to a run of 16 "ON" bits followed by an run of 16 uncompressed data bits.

- Solid Runs — The normal mode for IMG data is solid runs of RLE compressed data, read in bytes, so there is no tag to indicate the beginning of RLE data. Quite simply, a solid "ON" run is indicated by the high bit being set, and a solid "OFF" run is indicated by the high bit being off. The remaining seven bits indicate the length of the run in bytes. For example, a run of 24 consecutive "ON" bits is indicated by a byte value of 83 Hex, and a run of 32 consecutive "OFF" bits is indicated by a byte value of 04 Hex.

- Pattern Runs — Since any non-zero combination in the lower seven bits indicates a solid run, that leaves only the values 80 Hex and 00 Hex as indicators of non-solid runs of data. The 00 Hex value is used to indicate a pattern run, and the following byte defines the length of the run. The length of the pattern is defined in bytes 6 – 7 of the file header and that number of bytes following the run length define the pattern.

- Raw Data — Sometimes you'll need to encode bits in their raw form (for example, if you have a solid run less than eight bits long). For this purpose, the remaining tag (80 Hex) is used. The byte following the tag indicates the number of bytes of raw data that are to follow.

6 A word of caution when using raster files with Ventura Publisher: When you import a raster image into Ventura Publisher, it creates an IMG file and gives it the same name as the original. This is generally not a problem unless you import two images of different formats that use the same name. The IMG file created for the second image will overwrite the IMG file created (and used by Ventura Publisher) for the first image. If you are writing a program that creates IMG files, it may be nice to warn your users of this unexpected side effect.

TIFF

The Tagged Image File Format is a general purpose multi-platform raster format developed jointly by Aldus and Microsoft.

The Tagged Image File Format:

Developers' Desk
Aldus Corporation
411 First Avenue South
Seattle,, WA 98104 USA
Tel.206-622-5500

Windows Marketing Group
Microsoft Corporation
16011 NE 36th Way
Box97017
Redmond,, WA 98073-9717 USA
Tel.206-882-8080

Vital Statistics:

Raster Type
Very large max image size
Multiple images per file possible
Max color depth depends on class
Max palette depth depends on class
TIF extension

TIFF comes in many flavors. The default created by Graphics Converter is either black and white (if the source is monochrome) or grayscaled if the source is multicolored. This assures that most applications will successfully load a TIFF file created by Graphics Converter. These defaults can be adjusted using the Options dialog boxes. Also, if you are creating TIFF files for viewing, you should select *Force Gray* in the Advanced Option Dialog Box. *Force Black and White* produces better results for printing.

The TIFF specification is sophisticated and complex. Fortunately, its documentation is well written, well organized, and comes complete with background information and general examples. It is a must if you are writing

a TIFF reader or writer. The information below is intended to provide an overview and is not a substitute for the TIFF spec.

TIFF is probably the most versatile raster format available today — it is also correspondingly complex. TIFF files are designed without requiring the presence of any particular hardware or software. Rather than relying on defining the meaning of data by version number and position within the file, blocks of data are interpreted based on the tags that identify them. Thus, a program needs only to process data pointed to by meaningful tags. Such a structure is wonderfully malleable and the tag definitions are as rich as required by the broad scope of the TIFF specification. Unfortunately, this makes the creation of a *general* TIFF reader a formidable task.

TIFF was designed in 1986 specifically to provide a basis for importing scanned images into desktop publishing packages. At the time, scanner manufacturers produced their image files based on the manner in which their hardware produced data. Desktop publishing companies were more concerned with overall text layout than with supporting a whole band of proprietary raster formats. Thus, TIFF was designed to be powerful and flexible, but not simple, by any stretch of the imagination. As it turned out, all sorts of people started supporting, or at least claiming to support, TIFF because it was a sure-fire way to be compatible. Unfortunately, TIFF's designed flexibility allowed people to write all sorts of TIFF files which often made reading TIFF files a hapharzard affair. In order to provide a more usable set of guidelines, the developers of TIFF subdivided it into various functional groups, called classes, which are aligned roughly with the color model used in the file. Thus, though one program may create a perfectly valid TIFF file, that TIFF file may not be readable by another program claiming to support TIFF. However, even within TIFF classes, the rules are still quite loose. (TIFF is capable of supporting images of any color depth — even odd ones such as 3 or 5 bits per pixel.) This flexibility is great in that people aren't locked in or restricted, but it does make the job of digesting TIFF files somewhat difficult. In fact, there is probably no program that can read every valid TIFF file.

Why Tags?

As its name implies, TIFF is a tag based file format, which means that tags or identifiers are used to structure the raster data. This notion seems a bit unorthodox at first since raster data is generally thought of as a 2D matrix of pixel information for the entire image. Normally, tags are more readily associated with object oriented files, but in light of TIFF's original purpose, the tag concept makes good sense. As mentioned above, TIFF came about because of the unique needs of scanner and DTP manufacturers that provided several strong reasons for the tagged organizational structure:

- ☐ Scanners tend to create data in strips, so storing each strip as a unit made sense.
- ☐ Scanned files were considerably larger than conventionally created raster images and deemed too large to edit with graphics editors manipulating traditionally structured raster images. The tagged structure provided a framework for manipulating portions of images without reading and writing the entire image file.
- ☐ DTP people wanted scanned images to be viewable at screen resolution with a reasonable response time. This prohibited the processing of large data files. Tags provided a flexible structure in which multiple versions of an image could be maintained within each file.
- ☐ Such a structure allows application programs to look at the tags before deciding whether or not to process the data associated with them, allowing files to be less dependent on particular versions of software.

TIFF Classes

Currently, there are four and a half classes of TIFF.

Class B — *Bi-level images*. Simple Black and White (Single Bitplane) images usually produced by inexpensive scanners and suited for monochrome graphics devices such as the Hercules card and the Macintosh I.

Class G — *Gray scale images.* Multiple bitplane non-color images produced by mid-range scanners suited for more sophisticated graphics adapters (CGA, EGA, VGA).

Class P — *Color images using Palettes.* Multiple bitplane color images using an indexed color map to define color values. Generally created by graphics editing packages (though color scanners and video cameras may also produce files of this nature), these files also require some kind of color graphics adaptor.

Class R — *Full color images defining* R, G, B *values explicitly for each pixel.* Since these are the most sophisticated color graphics images supported by PC's, these files require a high level graphics adaptor (such as a TARGA board) for display and manipulation and a film recorder or other high end color output device for output.

Class F — FAX*images.* This is half class support. Essentially, fax card manufacturers took their existing file formats and added a TIFF tag either at the beginning or at the end of the file. Because of this, being Class F compatible doesn't guarantee complete compatibility with other software packages.

Compression

TIFF files can employ several types of data compression:

- ☐ Type 1 — (Uncompressed) No compression, other than squeezing as many pixels into a byte as possible. For color depths other than 2, 4, or 8, pad to nearest power of two.
- ☐ Type 2 — (Bilevel) Based on CCITT 1D compression that uses pre-defined tables to achieve compression. It is essentially the Group 3 Specification without padding or end of line codes.
- ☐ (Type 3) — Based on the CCITT Group 3 specification with padding and end of line codes.
- ☐ Type 4 — Based on the CCITT Group 4 specification.
- ☐ Type 5 — (LZW) Lempel-Zif and Welch compression.

- ☐ Type 32773 — (PackBits) A simple RLE scheme also supporting raw uncompressed data for complex patterns. This scheme is similar to those used by the IMG and WPG formats.
- ☐ Type *Guess?* — TIFF is flexible enough to allow for proprietary compression schemes other than those officially described in the TIFF specification. Files containing compression tags other than those above should not be expected to be processed by most applications, and there are some odd ones out there.

The TIFF Header

Due to the database-like structure of TIFF files, the header information (the identifier tags) is usually spread throughout the file. However, each TIFF file does have a simple header identifying it as such.

The TIFF Header

Byte #	**Description**	Notes and Values
0-1	Byte Order	4949H for Intel (LSByte to MSB) 4D4DH for Motorola (MSB to LSB)
2-3	Version Number	In Name only. Always set to 2AH.
4-7	Pointer to First IFD	IFD's begin on word boundaries.

- ☐ The byte order applies to the entire file and determines how the raw data is to be read before being interpreted. The Motorola microprocessors prefer their bytes ordered from the top down, whereas the Intel microprocessors prefer them bottom up.
- ☐ The version number is always set equal to 42, and since TIFF is version independent, the chances of this number ever changing are very small.
- ☐ Bytes 4-7 point to the first set of tags. (A set of tags is called an Image File Directory or IFD.) The IFD is simply a list of tags with

pointers to data and does not need to precede the actual data. Follow the pointers.

- ☐ Each IFD ends with a similar 4 byte field pointing to the next IFD, or is a field of 4 null bytes indicating that no further IFD's are present.

Thus, the first four bytes of a valid TIFF file should always be one of these two hexadecimal sequences:

49 49 2A 00 –OR– 4D 4D 00 2A

Beyond this, nothing in the TIFF file is location dependent. The last item in this header points to a table of tags that in turn either contain the image information or point to it. These tags are structured as follows:

The Image File Directory (IFD) Structure

Byte #	Description	Notes and Values
0-1	Entry Count	# of 12 Byte tag entries in directory. Note all addresses are relative to offset of IFD.
2-13	First Entry	12 Bytes long. Descriptions below.
14-25	Second Entry	12 Bytes long. Descriptions below.
12*Entries +2	Pointer to Next IFD	IFD's begin on word boundaries.

Notes All byte numbers shown above are relative to the location of the IFD within the file. These values are calculated by following the trail of pointers. The first pointer is in the simple file header, and each IFD ends with a pointer to the next IFD. The last IFD ends with a string of 4 null bytes to indicate that there are no further IFD's.

IFD's can be anywhere in the file. The data associated with them does not need to be near the IFD since each

entry has a 4 Byte pointer specifying the location of the data. IFD entries must be contiguous and should be sorted in ascending order by Tag ID number.

Each Directory entry is made up of 12 Bytes that are structured in the following way:

IFD Entry Structure

Byte #	Description	Notes and Values
0-1	Tag	Tag identification number. Defines how data is to be interpreted.
2-3	Type	1 = Byte (8 bit unsigned integer). 2 = ASCII (Null terminated 8-bit string). 3 = Short (16 bit unsigned integer). 4 = Long (32 bit unsigned integer). 5 = Rational (2 Longs,, Num/Den).
4-7	Count	Number of instances of Type. (Data Length = Type * Count).
8-11	Data or Offset	If the data can fit into 4 Bytes, put it here; otherwise this is a 4 byte pointer to the data.

Notes

The tag identifiers are listed in the TIFF specification although people add their own tags as needed. Some Tags support more than one data type, so never assume a type based on a tag.

The number of bytes occupied by the actual data associated with each tag must be calculated by multiplying the Type by the Count.

In cases where the tag data is four bytes or less, bytes 8 -11 contain the actual data rather than a pointer to it.

Organizing Raster Data under TIFF

Since raster images are created and viewed on many different devices, the way data need to be organized varies depending on the hardware being used. (Refer to the Planar vs. Non-Planar discussion under the IMG format.) In order to accommodate as many devices as possible and provide portability, TIFF constructs images based on the following terms.

- ☐ Image — A rectangular array of pixels
- ☐ Pixel — A dot having a color description made up of a number of samples.
- ☐ Sample — The number of elements needed to describe each dot.
- ☐ Bits per Sample — The number of bits needed to describe each sample.

Therefore, applications software must calculated the color depth of an image:

Color Depth = Samples per Pixel • Bits per Sample

The TIFF spec is full of "recommendations" about how files are to be written out. However, there are files where this color scheme is interpreted in strange ways. If possible, refrain from deviating too far from the recommendations in the TIFF specification. Graphics Converter compensates for various anomalies through the specification of flavors.

GIF

GIF is the image format owned and used by CompuServe. It is a sophisticated format providing the following capabilities:

- ☐ LZW compression
- ☐ Single and multiple images per file
- ☐ Interlaced and non-interlaced images.

Graphics Converter supports all these modes of GIF. If a GIF file contains multiple images and the destination type does not support multiple images, Graphics Converter will create multiple output files using the name of the source image as the basis for a numerically incrementing filename.

One of the richest collections of computer images is the CompuServe Picture Forum. If you are a CompuServe user, it is accessed by typing: GO PICS. If you are not a CompuServe user, it might be worth signing up just to access these images.

CUT

The CUT file format is created for a paint package called Dr. Halo. It is unique in that the palette information for the image data is not stored in a file header but in a separate palette file with a PAL extension. While this is useful if you need to switch palettes, it does present problems for file conversions.

When CUT is used as a Source type, Graphics Converter will attempt to find a palette file with the same name as the file being converted. Alternatively, you can specify a palette file using the Source Options dialog box. When writing out a CUT file, Graphics Converter will create a PAL file with a name to match the image file being created.

PIX

This is the native format used by InSet Versions 1 and 2 and Graphics Converter Version 1. It is used primarily for capturing screens.

- ☐ PIX files can contain bitmaps.
- ☐ PIX files can contain text screens.

If you have captured a text screen, you can create any output type from it. Otherwise, you can create only raster formats using PIX as your source. The Inset PIX format is used for graphics by WordStar and Multimate as well as Inset. The PIX format supports images up to 16 colors using the standard EGA and VGA palettes.

Vector Formats

The processing of vector files as input presents a unique challenge due not only to the variation in complexity from one vector format to another, but due also to the creative interpretation of standardized specifications within a format. Thus, the flavors problems associated with a raster format such as TIFF, are at least as prevalent in the vector domain with a format such as CGM. Finally, there are the problems associated with high level vector formats such as GEM 4 that accommodate complex curve forms, gradient fills, island moves, arbitrary polygon clipping, a multitude of fonts, and other sophisticated objects that must be resolved with controllable degrees of smoothness during a conversion to a simpler format.

DXF

The Document Interchange Format (DFX) is an Autodesk format for moving AutoCAD drawings to and from the rest of the world.

The Document Interchange Format

Autodesk, Inc.
2320 Marinship Way
Sausalito, CA 94965 USA
Tel.415-332-2344

Vital Statistics:

Vector type
ASCII encoded files
Used mainly for CAD drawings
DXF extension

Graphics Converter for Windows fully supports DXF including 3-Dimensional drawings. DXF files are normally ASCII encoded files, but there is an option with AutoCAD for producing binary encoded files in order to preserve space. Graphics Converter supports only the ASCII encoded version. Though these files can get quite large, Graphics Converter has successfully been tested with files which are several megabytes large.

- ☐ A floating point processor greatly improves the speed with which Graphics Converter processes these files.
- ☐ When converting a 3D image to a 2D image, Graphics Converter will use the last selected viewing direction.
- ☐ Graphics Converter supports 16 and 256 color DXF files having a fixed palette. 256 color DXF files can only be viewed correctly in AutoCAD using a PGA graphics adaptor. 16 color DXF files are mapped to 8 colors on a VGA display.
- ☐ DXF does not support filled polygons with more than 4 sides.
- ☐ Nested Parts are supported.
- ☐ There is no support for Shape files. (These files contain symbols.)

HPGL

HPGL (Hewlett Packard Graphics Language) describes plotter pen motion for the purposes of creating line drawings. Graphics Converter interprets these plotter commands as if it were a plotter.

- ☐ In Graphics Converter, HPGL support is based on emulating the HP7475A plotter.
- ☐ A special font called PLOTTER.TPF is provided for stroking out text contained in PGL files. Using this font ensures that the characters will look as though they were created by the plotter. PLOTTER.TPF is intended for used *only* with PGL Source files. Other formats, such as DXF, should use CAD.TPF to create the same look.

Color assignment is done by pen number rather than RGB specifications. Colors can be assigned in the Set File.

EPS

The Encapsulated PostScript format is a method of embedding postscript programs into information processed by applications software.

The Encapsulated PostScript Format:	Adobe Systems Inc. P.O. Box 7900 Mountain View, CA 94039 USA
Vital Statistics:	ASCII encoded files Vector and raster 24-bit color model Can contain multiple images Output only

PostScript is supported only as a Destination type.

- ☐ You can import these vector images into Adobe Illustrator (MAC or PC), CorelDraw, or Arts and Letters.
- ☐ Filled patterns are converted to RGB or gray levels.

GEM

The Digital Research "Graphical Environment Manager" (GEM) format is a native vector file format.

GEM Vector Format:	Digital Research Inc. P.O. Box DRI Monterey,, CA 93942 Tel.408-649-3896
Vital Statistics:	High level vector information A subset of GEM 4 GEM extension

GEM metafiles come in several flavors as shown in the table below.

Flavor	Max points per Polygon	Colors	Bezier
GEM	128	16 fixed	No
Artline V.2	128	16 w/dithering	Yes
Ventura	128	16 w/dither fills	Yes
Artline V.2	255	Edit 16 million	Yes
Presenta-	1024	View 16 mill.	Yes
tion, Ver. 2		Edit 16	

- ☐ When converting between flavors of GEM, Graphics Converter will break down polygons to accommodate the various limitations of a given package. Though this allows you to view images successfully, it makes editing them somewhat difficult. (Polygons are broken down so that maximum point totals are not violated.)
- ☐ Presentation Team images are generated in 16 colors since that is the number that can be manipulated in the editor.

CGM

The Computer Graphics Metafile (CMG) format is one of the most widely supported metafile formats.

CGM Metafile Format:	American National Standards Institute 1430 Broadway, New York,, NY 10018 USA Tel.212-354-3300
Vital Statistics:	An ANSI standard Medium complexity Both indexed and direct RGB color models supported CGM extension

The CGM format was created by the American National Standards Institute (ANSI) in order to provide a common ground for graphical information interchange. The CGM format is rapidly becoming the most widely supported format among major PC vector packages. This standard should permit the transportation of an image from one software application to another without loss or corruption of image content.

Because CGM is being supported by many people, a diversity of files is being created — all touting their CGM compatibility. Although in most cases the rules set forth in the ANSI CGM specificaton are not explicitly being violated, there are enough interpretations of the rules that user expectations (namely that a given CGM image should look the same regardless of the package being used to manipulate it) are not always met.

When Graphics Converter processes a CGM file as input, the input filter automatically detects the CGM flavor and compensates for the various

deviations produced by specific applications. When Graphics Converter produces a CGM file, these same idiosyncracies are written into the file so that the image will be properly processed by the application program specified as the output flavor. The areas in which some incompatibilities and exceptions exist are outlined below.

Bitmaps

As its name implies, it is a metafile format capable of supporting both vector objects and raster bitmaps in one file. In our experience, we have yet to see anyone put a bitmap (cell array in CGM speak) into a CGM file. Thus, Graphics Converter does not look for bitmaps when CGM is the source file type.

Pattern Fills

Graphics Converter does not support non-standard pattern fills. For example, some presentation programs allow you to create a pattern that looks like brickwork. If an image containing such a custom pattern were imported through the CGM input filter, none of the brickwork pattern areas would appear in the resulting image.

List of Operators			
CGM Operator	Class	ID	Supported
No_Op	00	00	Yes
Begin_Metafile	00	01	Yes
End_Metafile	00	02	Yes
Begin_Picture	00	03	Yes
Begin_Picture_Body	00	04	Yes
End_Picture	00	05	Yes
Metafile_Version	10	01	Yes
Metafile_Description	10	02	Yes
VDC_Type	10	03	Yes
Integer_Precision (16 Bits)	10	04	Ignored
Real_Precision	10	05	Ignored
Index_Precision	10	06	Ignored
Color_Precision (<= 16 Bits)	10	07	Yes
Color_Index_Precision	10	08	Yes

Maximum_Color_Index	10	09	Yes
Color_Value_Extent	10	0A	Yes
Metafile_Elements_List	10	0B	Ignored
Metafile_Defaults_Replacement	10	0C	Ignored
Font_List	10	0D	Yes
Character_Set_List	10	0E	Ignored
Character_Coding_Announcer	10	0F	Ignored
Scaling_Mode	20	01	Ignored
Color_Selection_Mode	20	02	Yes
Line_Width_Specification_Mode	20	03	Yes
Marker_Size_Specification_Mode	20	04	Yes
Edge_Width_Specification_Mode	20	05	Yes
VDC_Extent	20	06	Yes
Background_Color	20	07	Yes
VDC_Integer_Precision	30	01	Ignored
VDC_Real_Precision	30	02	Ignored
Auxiliary_Color	30	03	Ignored
Transparency	30	04	Ignored
Clip_Rectangle	30	05	Yes
Clip_Indicator	30	06	Yes
Polyline	40	01	Yes
Disjoint_Polyline	40	02	Yes
Polymarker	40	03	Yes
Text	40	04	Yes
Restricted_Text	40	05	Ignored
Append_Text	40	06	Ignored
Polygon	40	07	Yes
Polygon_Set	40	08	Ignored
Cell_Array	40	09	Ignored
Generalized_Drawing_Primitive	40	0A	Ignored
Rectangle	40	0B	Yes
Circle	40	0C	Yes
Circular_Arc_3_Point	40	0D	Yes
Circular_Arc_3_Point_Close	40	0E	Yes
Circular_Arc_Center	40	0F	Yes
Circular_Arc_Center_Close	40	10	Yes
Ellipse	40	11	Yes
Elliptical_Arc	40	12	Yes
Elliptical_Arc_Close	40	13	Yes

Line_Bundle_Index	50	01	Ignored
Line_Type	50	02	Yes
Line_Width	50	03	Yes
Line_Color	50	04	Yes
Marker_Bundle_Index	50	05	Ignored
Marker_Type	50	06	Yes
Marker_Size	50	07	Yes
Marker_Color	50	08	Yes
Text_Bundle_Index	50	09	Ignored
Text_Font_Index	50	0A	Yes
Text_Precision	50	0B	Ignored
Character_Expansion_Factor	50	0C	Ignored
Character_Spacing	50	0D	Ignored
Text_Color	50	0E	Yes
Character_Height	50	0F	Yes
Character_Orientation	50	10	Yes
Text_Path	50	11	Yes
Text_Alignment	50	12	Yes
Character_Set_Index	50	13	Ignored
Alternate_Character_Set_Index	50	14	Ignored
Fill_Bundle_Index	50	15	Ignored
Interior_Style	50	16	Yes
Fill_Color	50	17	Yes
Hatch_Index	50	18	Yes
Pattern_Index	50	19	Ignored
Edge_Bundle_Index	50	1A	Ignored
Edge_Type	50	1B	Yes
Edge_Width	50	1C	Yes
Edge_Color	50	1D	Yes
Edge_Visibility	50	1E	Yes
Fill_Reference_Point	50	1F	Ignored
Pattern_Table	50	20	Ignored
Pattern_Size	50	21	Ignored
Color_Table	50	22	Yes
Aspect_Source_Flags	50	23	Ignored
Escape	60	01	Ignored
Message	70	01	Ignored
Application_Data	70	02	Ignored

Graphics Converter Error Messages

InsufficientMemory: The program has run out of memory. Close some program windows and try again.
File Name Already Exists: That file does not exist. If you are using a network, perhaps your access status is too low.

Could Not Write To File: The file is locked or an error occurred while writing. Perhaps there isn't enough space on the disk, or your floppy disks are write protected.

Could Not Read A File: A disk error occurred during a read operation. The input file may be corrupt, or the file format is unknown.

Unexpected End of File Occured: The input file is probably corrupt or is not a recognized format.

General File Error: The input or output file is probably corrupt.

The Tag File is Probably Corrupt: The tag file contains unrecognized commands. Review the contents of the tag file according to this manual.

An Error Occurred Writing to a Tag File: The target disk is full or write-protected.

Couldn't Open the Tag File: The specified tag file cannot be found or your network status is too low.

Couldn't Create the Specified Tag File: The filename is illegal or your network status is too low.

Invalid Tag File: The tag file contains unrecognized commands. Review the contents of the tag file according to the documentation.

Filename Too Large: The specified filename is longer than eight letters.

File Spec Not Valid: The specified filename is longer than eight letters.

Drive Is Not Ready: If the drive is a floppy disk, make sure that the drive door is closed and / or a formatted disk is in the drive.

Path Not Found: That path doesn't exist. Try another path.

File Not Found: That file doesn't exist. Try another file.

Error Opening Source Files: The file does not exist or your network status is too low.

Error Opening Destination File: The file is protected on the network.

Source File Type is Unsupported: The type is not supported.

Source File Doesn't Match Type: The file is corrupt or not the specified type.

Destination File Type is Unsupported: That file type is not supported in the current version.

General Convert Library Error: An error occurred. The source may be corrupt. Try again.

Miscellaneous Error: This generally indicates a corrupt file.

Set Files

Graphics Converter option settings are contained in a "SET" file called WINCVT.SET. These settings are given in the form of ASCII text strings where each tag has a name and value. In a SET file, each tag is on it's own line and takes the form of "NAME=VALUE" where the tag NAME is to the left of the '=' and VALUE string is to the right of the '='. There should be no space characters between the NAME, '=', and VALUE strings. The name and value strings are not case sensitive but for consistency sake they should be uppercase.

Something that is used in most of the convert tags is file format codes. A file format code is a three letter string that unique for each format. In most cases it is the standard file format extension used in it's DOS file names (Example: PCX for PC Paintbrush, GIF for Compuserve GIF). Where file extensions conflict with existing formats a unique 3 letter extension will be used.

The types of convert option tags can be broken down into the following categories:

Operation Tags

These are options controlling general conversion operation such as debug information, buffer sizes etc. These tag names generally don't contain any format codes (Example: "DEBUG=ON" to turn on the generation of debug files).

Generic Format Tags

These are options are common across most formats but are used as the specification for a particular source or destination file format. They always start with the format code for a particular type and follow with a generic option string (Example: "PCXW=5.0" creates a PCX file 5 inches wide).

Generic Format Pair Tags

These options are common across most formats but are used as the options specification for a particular source/destination file format pair. They are formed by using source format code followed by the destination format code followed by the option code (Example: "CGMPCXINV=ON" inverts the output in CGM to PCX conversions. CGM = source format code, PCX = the destination format code, INV = the option code, and ON = tag value).

Specific Format Tags

These tags are similar in form to the Generic Format Tags except that they are not generally applicable across most formats. They take the same form as the Generic Format Tags (Example: "PCLFF=ON" to put a form feed in PCL output).

Common tag values, and the nomenclature used to specify them in this document, generally fall within the following categories:

##	=	Integer whole numbers
##.##	=	Real Number Values
RRGGBB-RGB	=	Value where each color is specified in 2 hexadecimal digits (Example: FF0000 specifies red)
ON, OFF	=	Used to turn an option on or off

Operation Tags:

The following tags control general internal conversion operations:

AliasExt=FileType

This tag controls the mapping of file extensions to 3 character file types. While in most cases the Inset file type is the same as the normal file extension, this isn't always the case. For example, if the .GMF extension is used as an alias for the CGM type the tag would be: GMF= CGM

TEMPPATH=Path

Sets the path to the directory that will be used for temporary files.

BUFFERSIZE=##

Controls the size of the raster tiles when performing raster to raster and vector to raster processing. The default is 32768 bytes and the range is from 4096 to 65536. Example: "BUFFERSIZE=16384"

DEBUG=ON,OFF

Controls whether internal meta operator files are saved after the conversion. The default is OFF (which deletes the files). Example: DEBUG=OFF

HEAPSTAT=ON,OFF

Displays the maximum system heap usage to help determine the memory requirements. The default is OFF (the heap usage will not be displayed). Example: HEAPSTAT=OFF

MEMCHECK=ON,OFF

Controls the enabling of internal diagnostic checks for memory not being freed during conversions. The default is "MEMCHECK=OFF"

Generic Format Tags

The following tags control input options. The actual tag for a specific format is formed by replacing the "sss" string with the format code for the source format.

sssSFsrcfontname=isifontname

This tag controls the mapping of a source font to an internal ISI font. Substitution strings are (Example: DXFSFHELVETICA=HELV):

srcfontname — Name of native font as recognized in the source file

isifontname — Name of ISI internal font which either represents a .TPF file or a user defined internal font.

The following tags control output options. The actual tag for a specific format is formed by replacing the "ddd" string with the format code for the target format.

dddDFisifontname=dstfontname

This tag controls the mapping of an internal ISI font to a native destination font. Substitution strings are (Example: CGMDFHELV=HELVETICA):

dstfontname — Name of native font as recognized in the destination format

isifontname — Name of ISI internal font which either represents a .TPF file or a user defined internal font.

dddDEP=##

Controls the output depth in terms of number of bits deep. Defaults to depth of source if not specified.(Example: PCXDEP=1)

dddW=##.##

Controls the width of the output image in inches. (Example: PCXW=5.0)

dddH=##.##

Controls the height of the output image in inches. See sizing rules if not specified. Example: PCXH=3.0

dddA=##.##

Controls the aspect ratio of the output image in fractional terms of width divided by height. See sizing rules if not specified. (Example: PCXA=1.0)

dddVR=##

Controls the vertical resolution of the output image in decimicrons (10 to the -10th meters). The default is usually 846 (300 dpi). (Example: PCXVR=846)

dddHR=##

Controls the horizontal resolution of the output image in decimicrons (10 to the - 10th meters). The default is usually 846 (300 dpi). (Example: PCXHR=846)

dddSFREQ=###

Controls the screen frequency in terms of decimicrons of halftone dots when color processing is set to HALF. The SFREQ value should be greater than the resolution values. The default value is 4320 (60 lines per inch).(Example: PCXSFREQ=4320)

Sizing Rules

- If no output size is specified then the output width is 5 inches.
- If only width is specified then the height is generated by using the width and maintaining the aspect ratio. Similarly, if only the height is specified the width is generated maintaining the aspect ratio.
- If height or width is defined and aspect ratio is not specified it is 1.0 (or 1 to 1). If both height and width are specified and aspect is not, then the aspect ratio is stretched to fill the frame.
- If height, width and aspect ratio are specified the image is scaled to fit within the frame.

Generic Format Pair Tags:

The following are source/destination pair processing options. The actual tag for a pair is formed by replacing the "sss" and "ddd" strings with the format codes for the source and destination.

sssdddINV=ON, OFF

Controls black & white swap (Example: "PLTPCXINV=ON")

sssdddCOG=C, G, B

Controls color conversion converting source colors to:

C = Color (default)

G = Gray values using an algorithm to match black and white television results.

B = Snaps colors to either solid black or white (Example: PLTPCXCOG=G to produce gray output from PLT to PCX).

sssdddBSWAP=ON,OFF

Controls whether a background/foreground color mapping is performed (the default is on). (Example: CGMPCXBSWAP=ON)

Format Codes and Format Specific Tags:

The following is a listing of format codes and any format specific tags associated with them.

Vector File Types:

HPGL - Hewlett Packard Plotter

PGLP#=RRGGBB

This tag assigns a specific color (in RGB) to a pen number. The number (#) can be from 0 to 15, and the RGB definitions are in hex. (Example: PGLP3=FF00FF sets pen 3 to magenta.)

PGLW#=penwidth

This tag assigns a width to a specific pen. (Example: PGLW3=5 sets the width of pen three to be three units wide.)

CGM - Computer Graphics Metafile

CGM16COL=ON,OFF

This tag determines whether a CGM file will be written with RGB colors or a palette of 16 colors. Default is OFF (RGB color selection).

CGMFLAV=Freelance Harvard Graphics Word Perfect

This tag sets the flavor of CGM to be written. Anything can go here but the above three are the only ones recognized. Default is generic CGM (none of the above).

GEM - GEM Metafile

GEMFLAV=Draw PerFORM Artline 1.0 Ventura Artline 2.0 Pres. Team 2.0

This tag sets the flavor of GEM to be written. The above are the only flavors recognized. The default is Draw.

Raster File Types

CUT - Dr. Halo

CUTOPAL=PalFileName

This tag specifies the name of the file to be generated that contains the palette information. If not specified, a file with the same name as the destination with a PAL extension will be generated.

GIF - Compuserve Gif

GIFINT=ON,OFF

This option controls whether or not the GIF file will be interlaced. The default is OFF.(Example: GIFINT=ON)

TIF - Tagged Image File Format

TIFTYPE=FlavorFormula

This tag controls the formula for the current TIFF type to be generated. A Flavor formula consists of one or more depth formulas separated by semi-colons.

CPU DepthFormula [;DepthFormula[...]]

Either an I for Intel or M for Motorola

Each Depth formula is broken down as follows:

Depth Class Compression [+tag,type,data...] [-tag]

Where: Depth is the pixel depth for this formula

Class is the TIFF Class type (B,P,G,R,F)

Compression is one of the following types:

1 = None

2 = CCITT G3 with no EOL

3 = CCITT G3 including 2d

4 = CCITT G4

5 = LZW

6 = 32773 (Mac packed bits)

\+ Tag - Adds a TIFF tag of the following spec tag - is the TIFF tag # type - Specifies one of the following data types:

1 = Byte
2 = ASCII (each byte = 1 count)
3 = SHORT
4 = LONG
5 = Rational (Data as 2 longs = 1 count)
count - Number of data items of the specified type follow
data - comma separated decimal encoded numbers of the size specified by type or a string in the case of type ASCII
- Tag - Removes a TIFF tag from the output

(For example, to generate a FlavorFormula for a type that can accept Intel TIFF with depths of 1 as Class B with G3 compression and a depth of 4 as Class G with no compression: TIFTYPE=I1B2;4G1)

TIF_FlavorName=FlavorFormula

This is an interactive tag that allows additional flavors to be added to the TIFF destination options box where "FlavorName" is the name of the flavor that appears in the TIFF flavors box. When a particular flavor is selected, the FlavorFormula string is assigned to the TIFTYPE tag.

Note: Since the list box to display the application name is 16 characters wide, the FlavorName should be limited to a maximum of 16 characters.

Mixed Vector/Raster Formats

EPS - Encapsulated Postscript

EPSVIEW=OFF,TIF

This controls whether or not a viewable header is included. The default is OFF. Example: EPSVIEW=TIF

EPSFF=ON,OFF

This controls whether or not a page eject is inserted into the file. The default is OFF.

EPSTYPE=EPS,AI

This option controls the flavor of EPS file generated. The default is EPSTYPE=EPS

EPSMPAGE=ON,OFF

This option controls whether or not output from a multipage input source will be put in a single file (EPSMPAGE=ON) or multiple files (EPSMPAGE=OFF).

EPSMODEL=RGB,CMYK

This option controls the color model used in EPS output. RGB is the default (EPSMODEL=CMYK)

EPSUCR=###.###

This option controls undercolor removal when EPSMODEL=CMYK. The value is a floating point number in the range from 0.0 to 1.0 (default is 1.0). The value is used to remove Black from the Cyan, Magenta, and Yellow colors. The algorithm works as follows:

```
if ((C=EPSUCR) AND (M=EPSUCR) AND (Y=EPSUCR)) THEN
        C = C - EPSUCR
        M = M - EPSUCR
        Y = Y - EPSUCR
        K = EPSUCR
ELSE
        K = 0 (No Black)
```

GRAPHICS CONVERTER

TOTAL GRAPHICS FLEXIBILITY FOR IMAGE CONVERSION

Graphics Converter allows you to swiftly convert any DXF or HPGL file to several different raster and vector file formats. Supported file types are PCX, TIFF, GIF, IMG, CUT, PIX, GEM, CGM and EPS.

System Requirements:

286, 386, 486 100% IBM compatible
Windows Version 3.0 or greater
A hard drive with 2 megabytes of free space
Either a 5.25" or 3.5" floppy drive
2 megabytes of RAM
A math co-processor is recommended for DXF conversions

Imsi™
1938 Fourth Street • San Rafael, CA 94901
Tel: 415 454-7101 • 800 833-4674
Fax: 415 454-8901 • BBS: 415 454-2938

IMSI (UK) Limited
Unit 17 Brook Lane Business Centre
Brentford, Middlesex, TW8 OPP, U.K.
Tel: (081) 758 1447 • Fax: (081) 758 1667